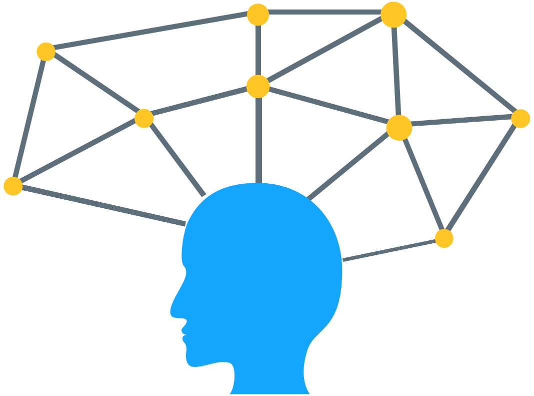
DSA5101
Machie Learning Project

By Huang Xijie, Li Zitian, Wang Shuhui

Github Link: https://github.com/lzt68/2021-DSA5101-Machine-Learning-Project

Content

- 1. Overview
- 2. Data Exploration and Preprocessing
- 3. Model Selection Feature Engineering
    - Huang Xijie: XGboost vs LightGBM
    - Li Zitian: Random Forest vs Decision Tree
    - Wang Shuhui: Logistic Regression vs Knn
- 4. Insight
- 5. Future Direction



Overview

Overview

Data

- **Marketing campaign** of Portuguese bank
- **Imbalanced** binary label

Measure

- **MCC** score
- Our score **0.61**

Preprocess

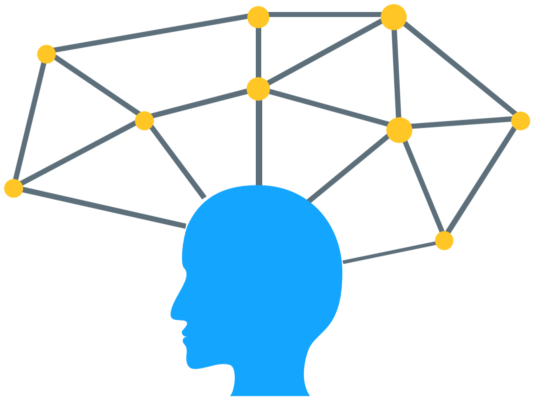
- **Resample**
- Add **timestamp**
- **Forward** selection
- Feature **Importance**

Algorithm

- **XGBoost**
- Random Forest
- Logistic Regression
- KNN
- LightGBM
- Decision Tree

Insight

- **Exogenous** factor matters
- **Duration** of contact matters
- **Continuity** of clients' behavior



Data Exploration Preprocessing

Data Exploration

Label

- **Binary** value
- ‘no’: 39922 88%
  ‘yes’: 5289 12%
  Imbalanced

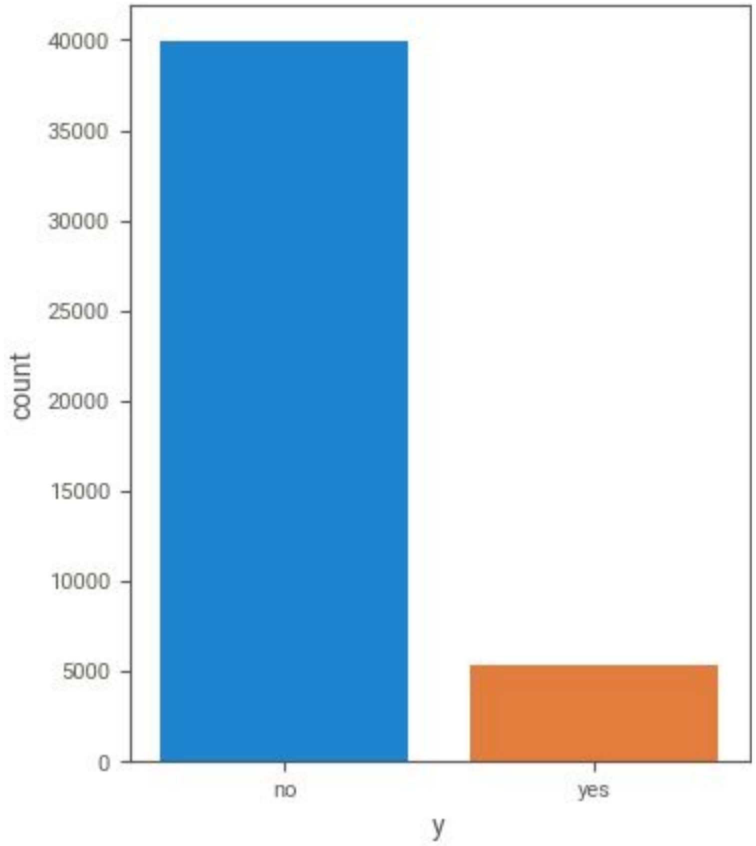
Numerical

- **7** numerical features
- 'age', 'balance', **'duration'**, days
  'campaign', 'pdays', 'previous'
- **Heavily right** tailed

Categorical

- **8** categorical features
- 'job', 'marital', 'education',
  'default', **'housing'**, 'loan',
  'contact', **'poutcome'**,
  month
- **Imbalance** also exists

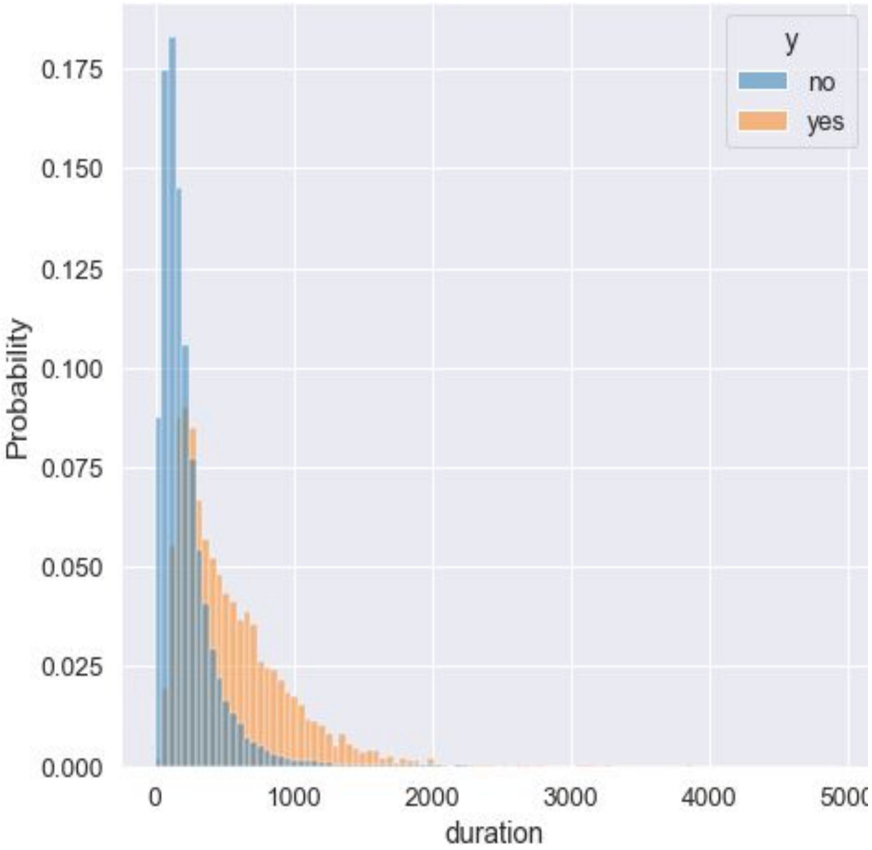
Data Exploration



Lable: ‘y’

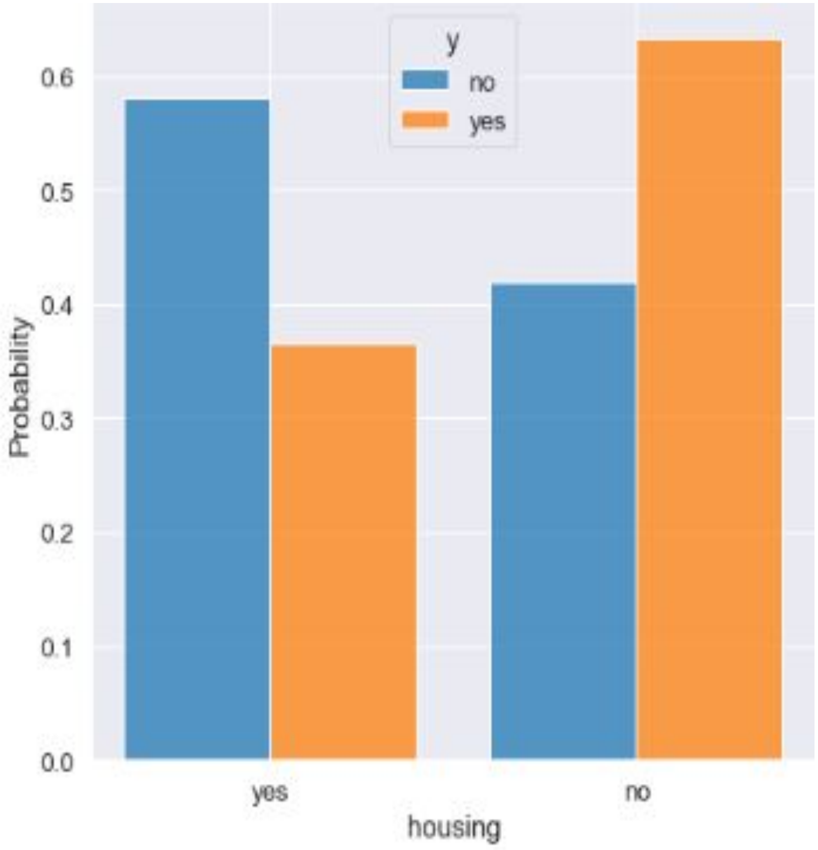
- **Success or Failure of** contact
- **Inbalanced** binary label

Data Exploration



duration

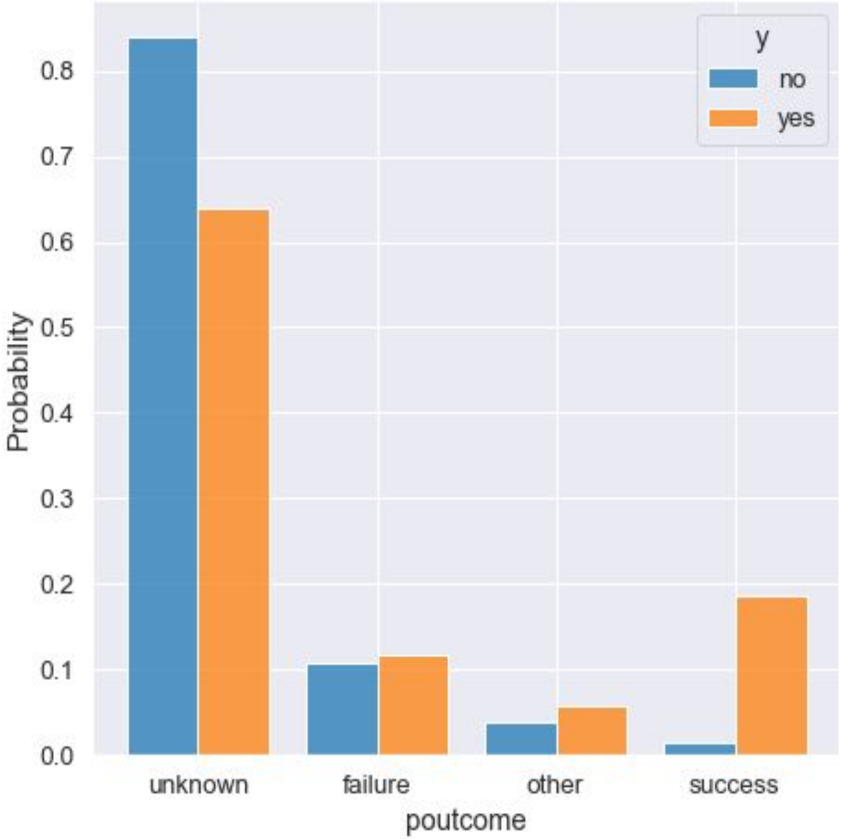
- **Length of last contact**
- client's **interest**
- higher duration infer **higher successful rate**



housing

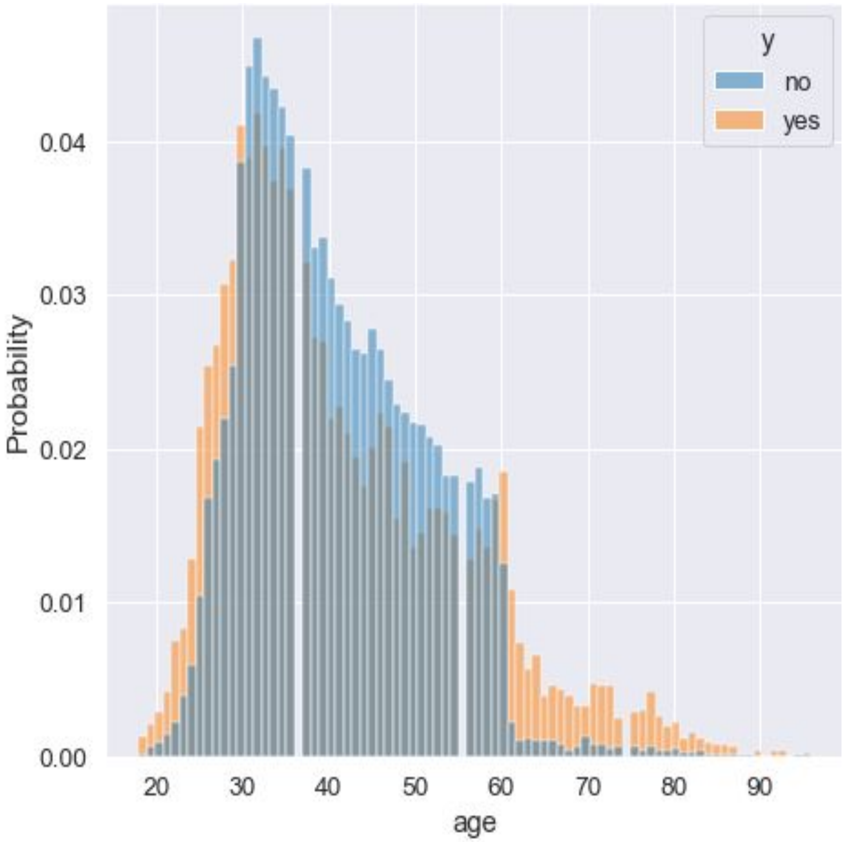
- Existence of housing **loan**
- might affect **disposable income**
- Housing loan may **decrease** successful rate

Data Exploration



poutcome

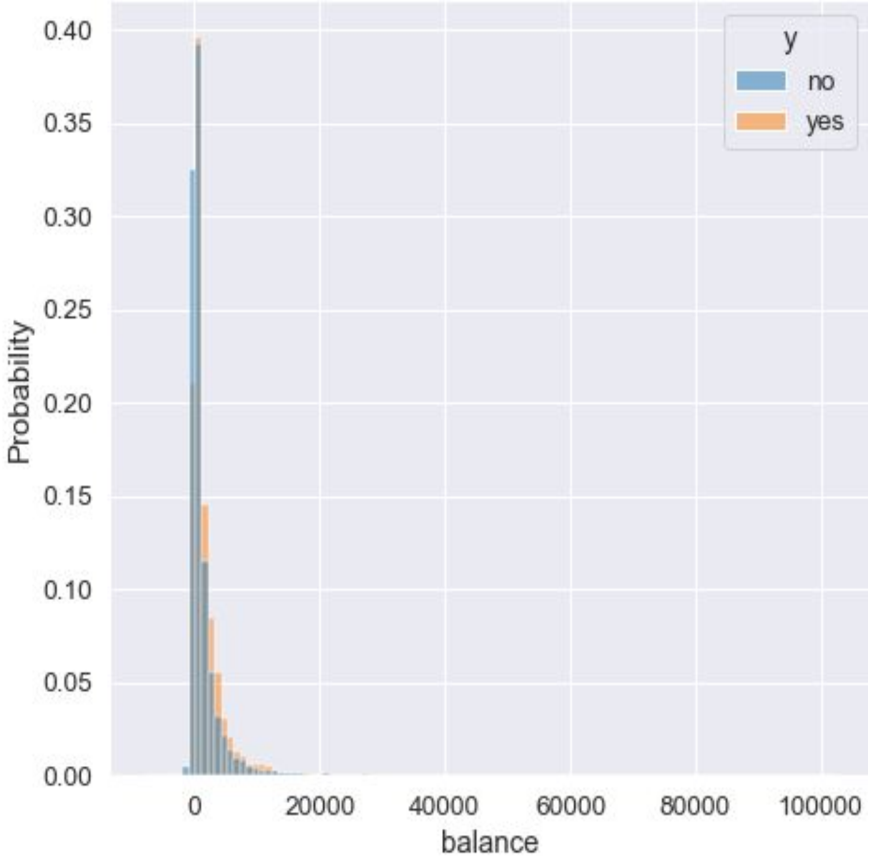
- Result of **last** contact
- **Inbalanced** feature
- Success in the last time may infer **higher** successful rate



age

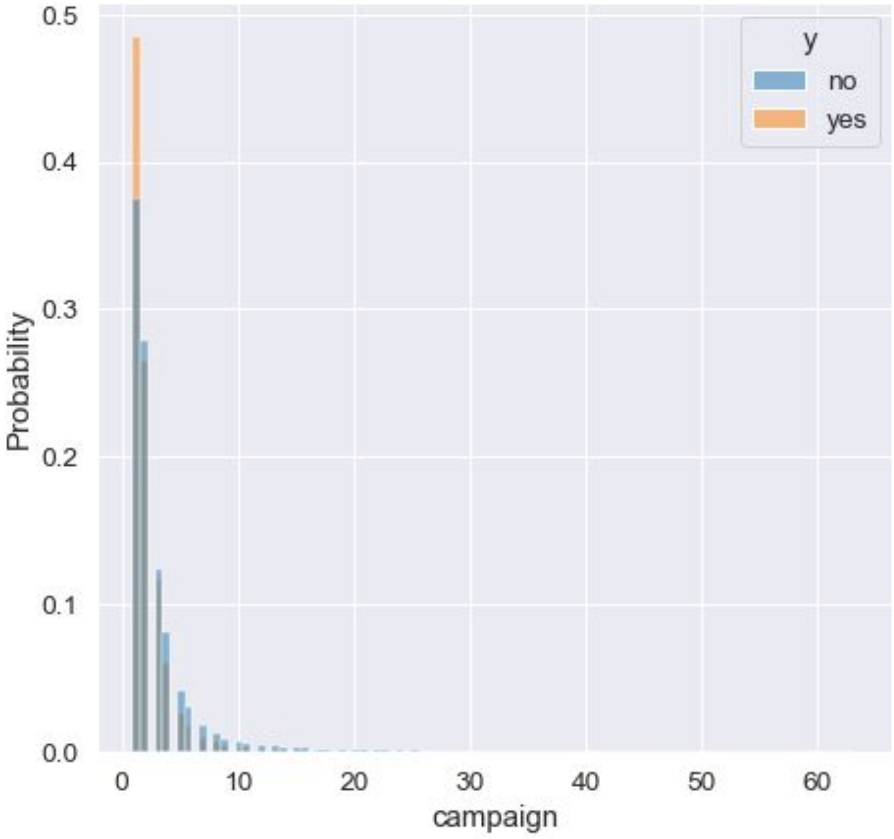
- Range from 18 to 90
- Most of clients are middle-aged
- Middle-aged clients are more likely to **reject** term deposit

Data Exploration



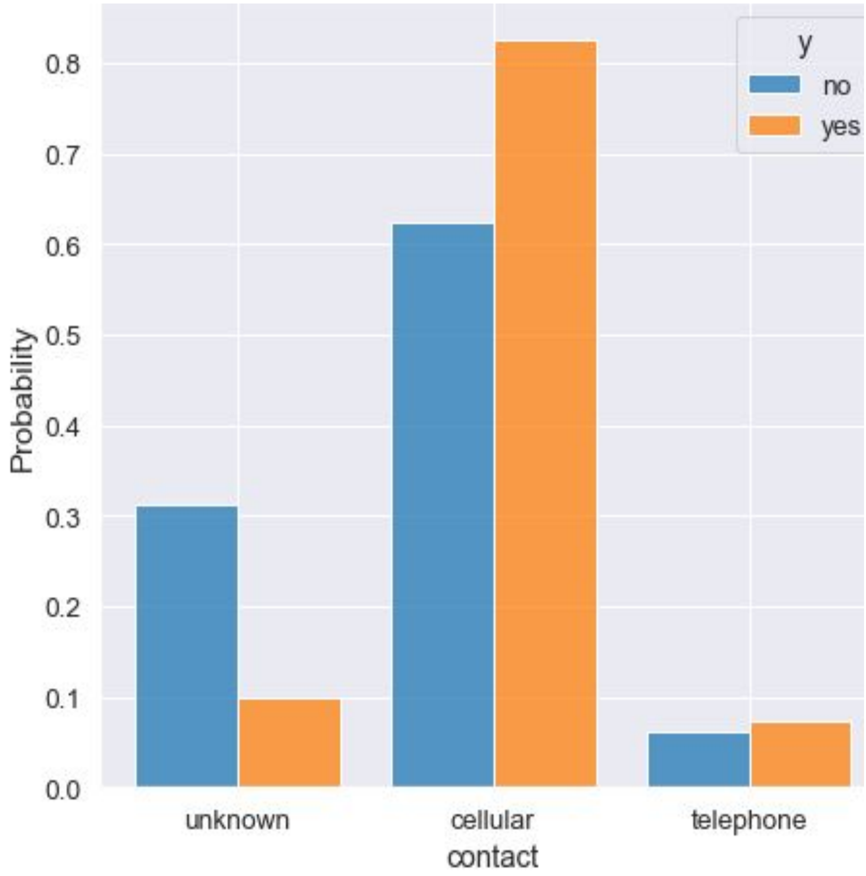
balance

- Some are **indebt**
- **Not heavy-tailed**



campaign

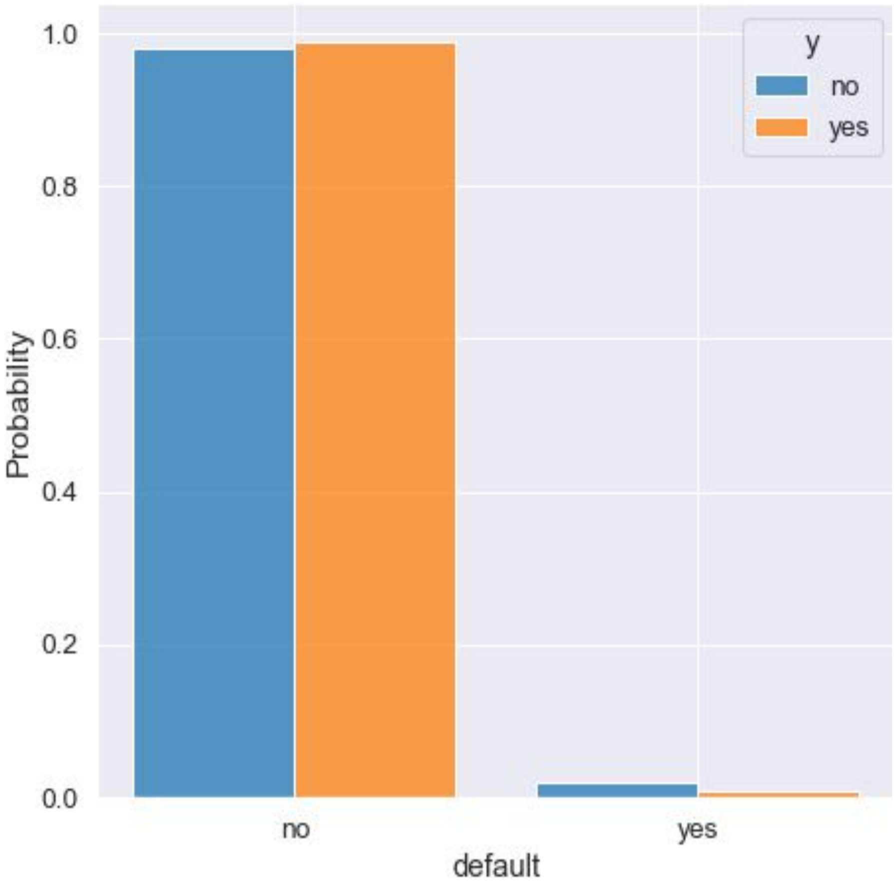
- Number of performed contacts
- Clients with **less** contacts seem more declined to term deposit
- **Not significant** in the models, compared to other features



contact

- Communication type
- Cellular seems related to higher success rate
- **Not significant** in the models, compared to other features

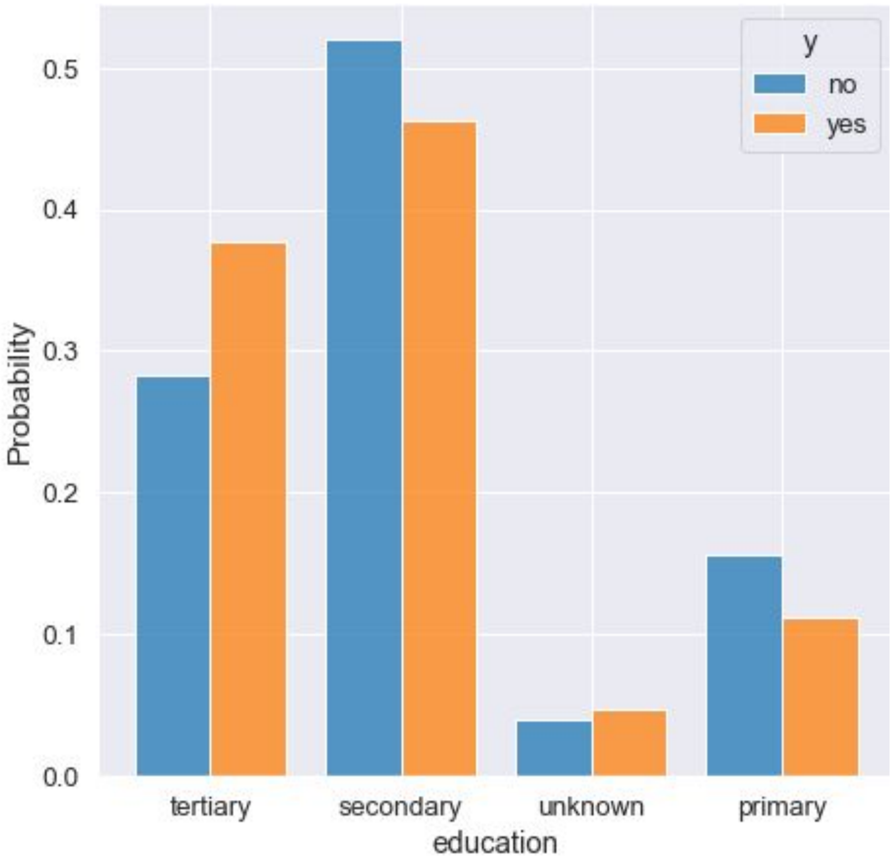
Data Exploration



default

- Credit records
- highly **imbalanced**
- **Cannot derive** much inofrmation directly from this feature

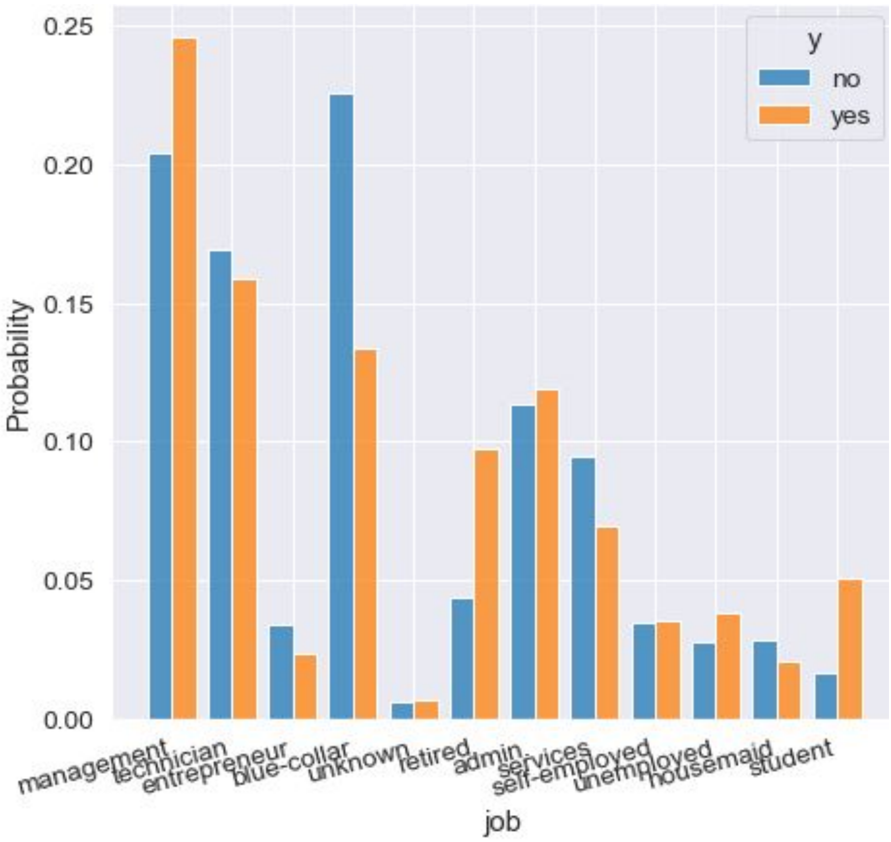
Data Exploration



education

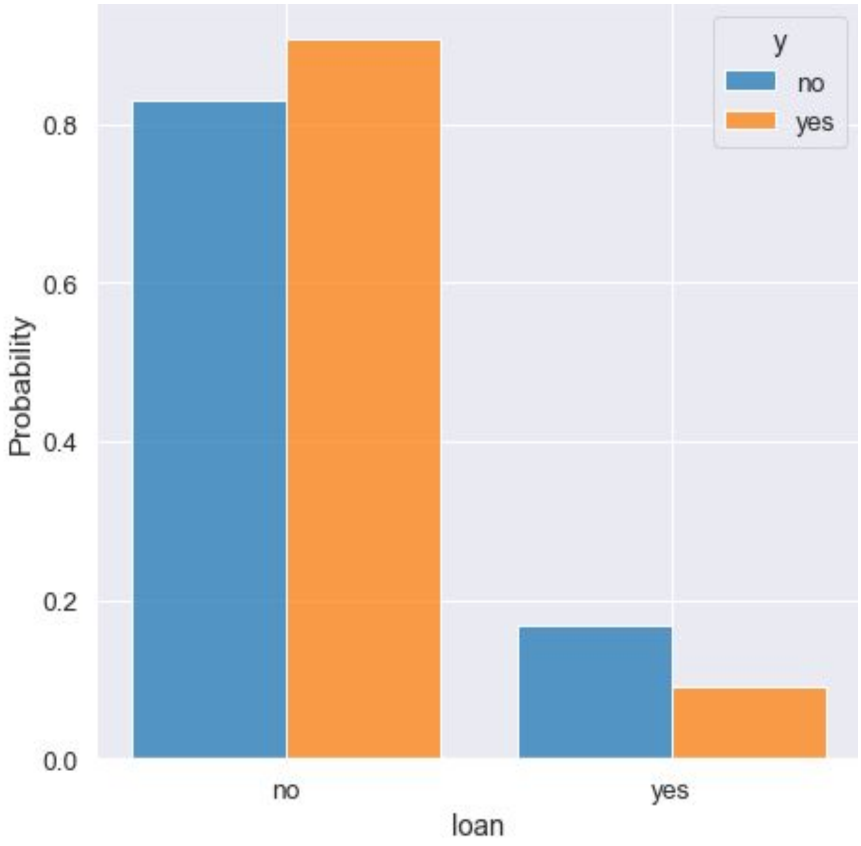
- Educational level
- Higher education background seems to prefer term deposit

Data Exploration



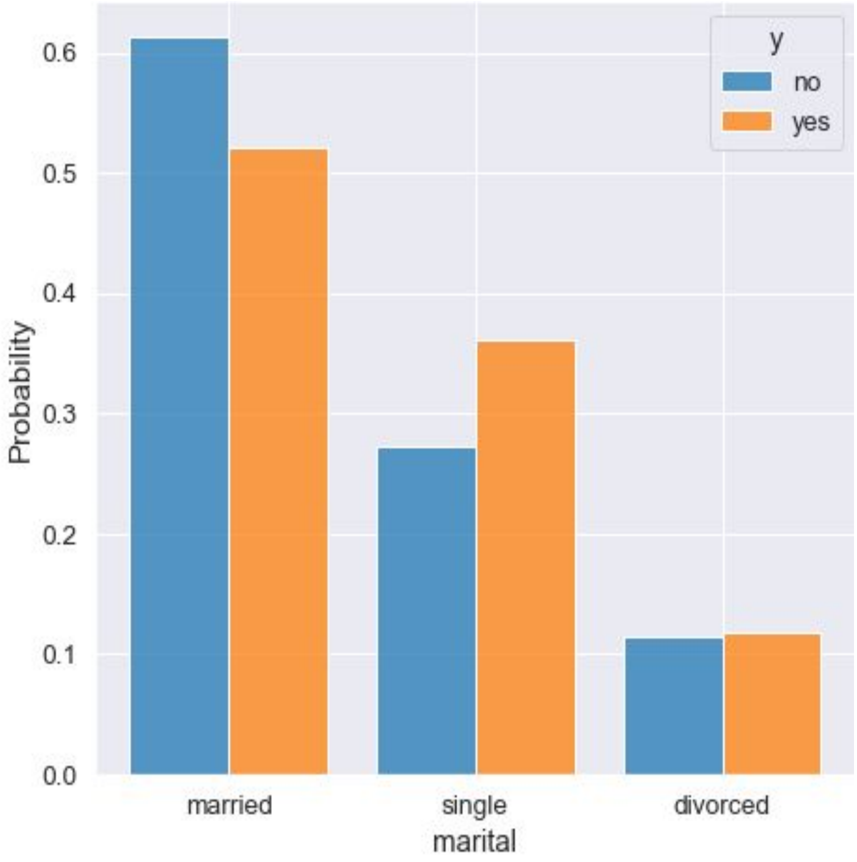
job

- Multiple values
- **Irregular relationship** with successful rate
- **Further encoding needed**



loan

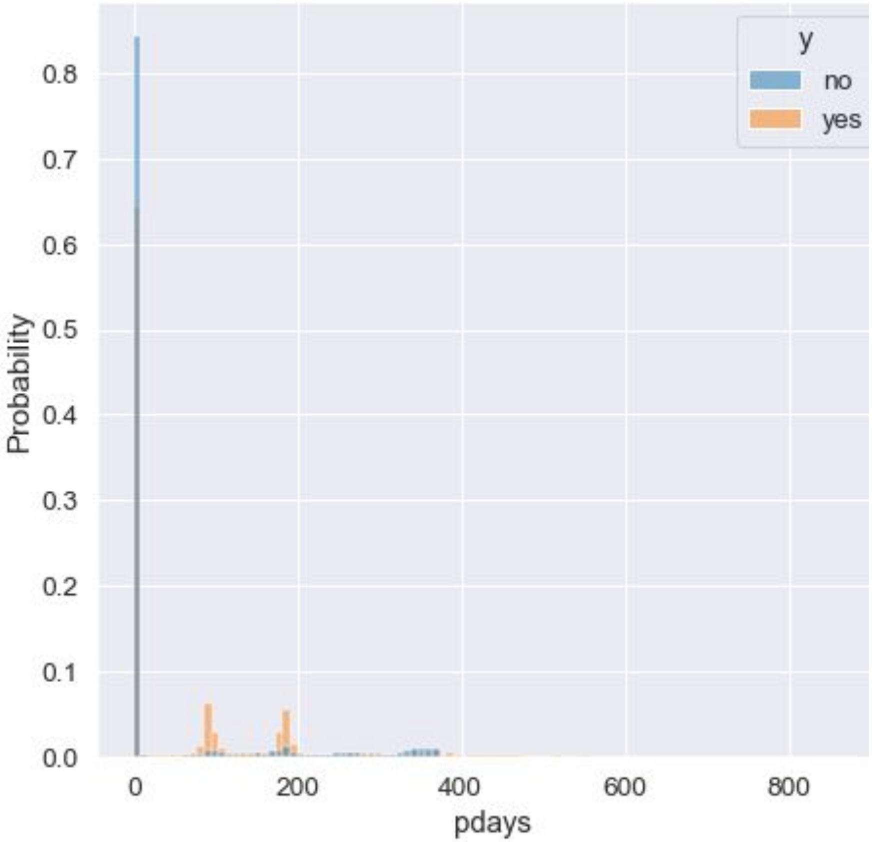
- Personal loan
- To some extent, reflect the financial condition of clients
- **Not significant** in the models, compared to other features



marital

- Marital condition
- **Single** clients may be more inclined to accept term deposit, but **not highly significant** in models

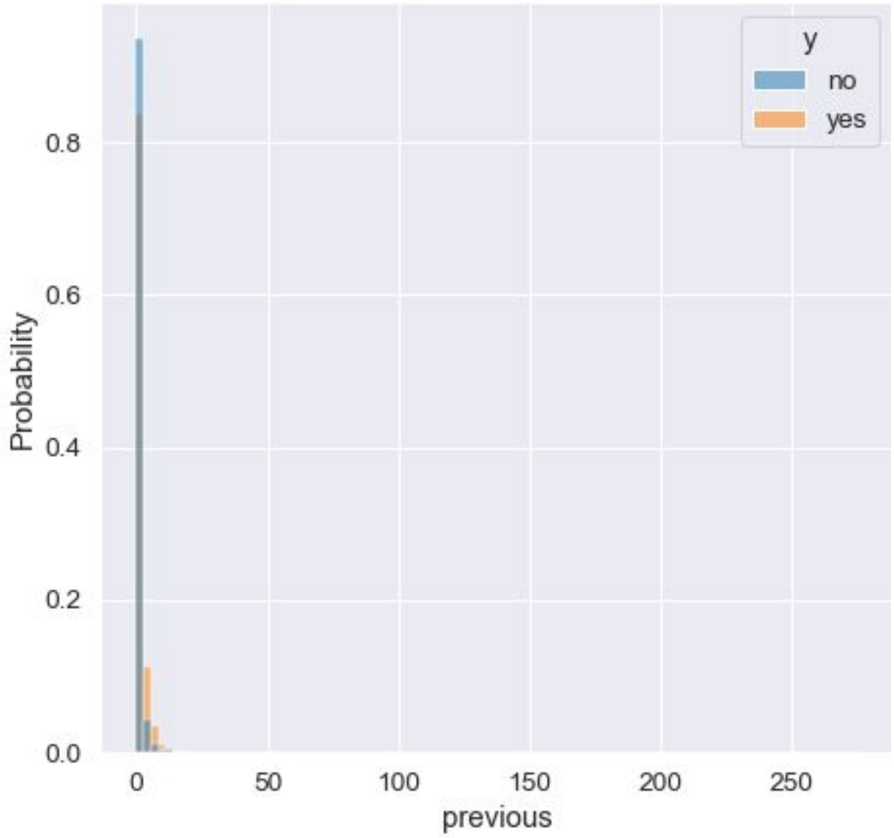
Data Exploration



pdays

- Most of clients had never been contacted
- **Further encoding** can make this feature useful, but not as important as others

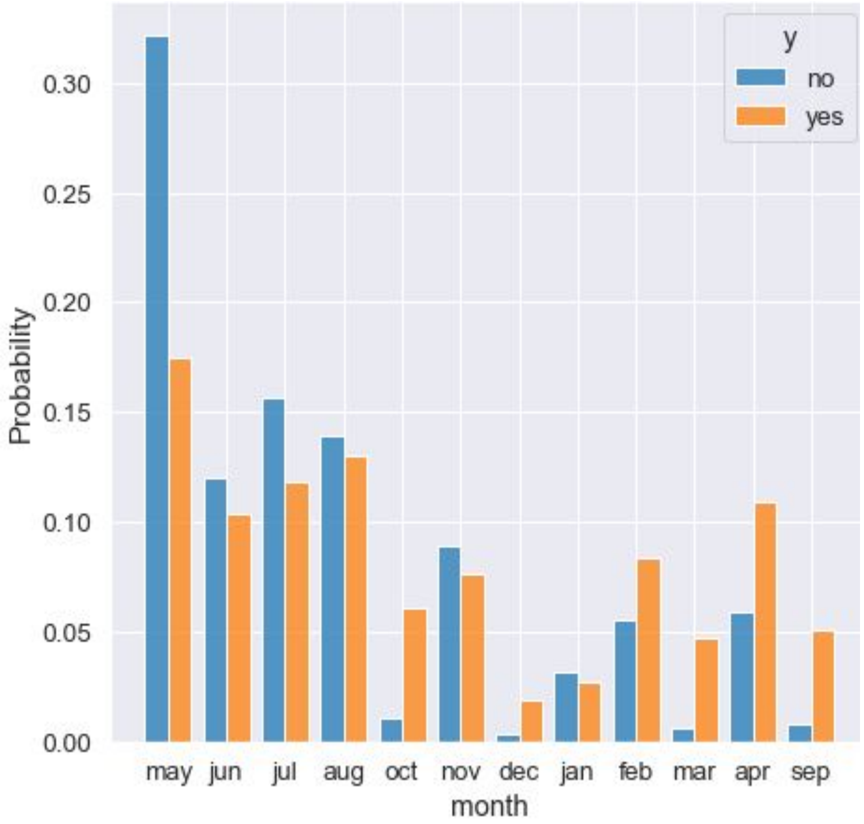
Data Exploration



previous

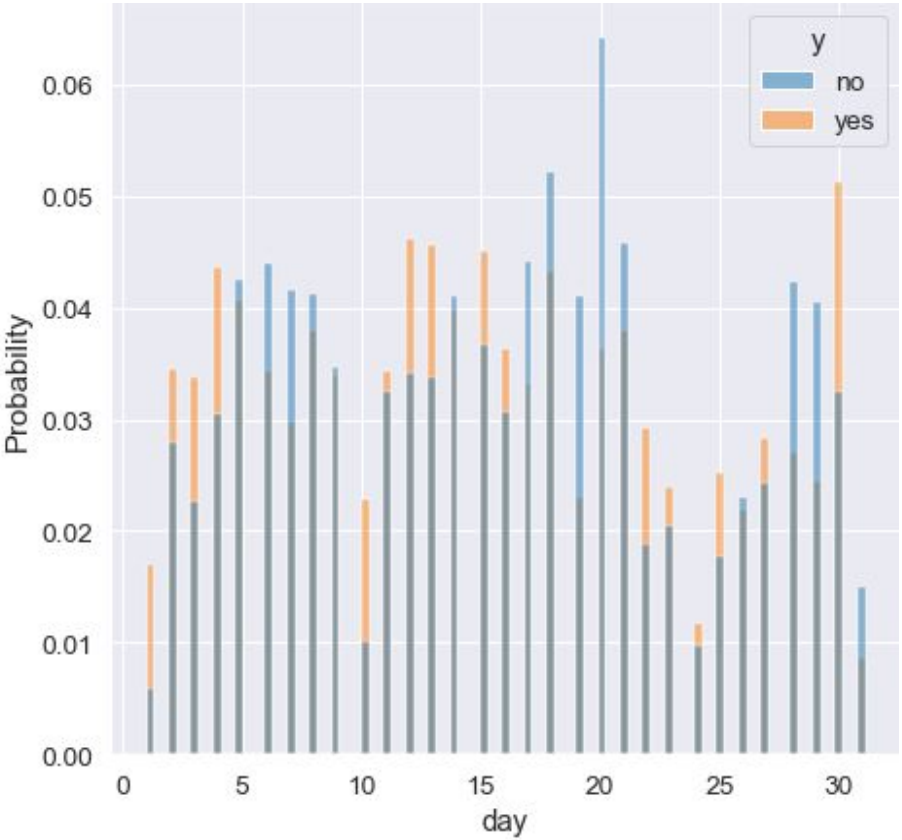
- Highly imbalanced
- It might be useful in **specific samples**, but not as important as others

Data Exploration



month

- Information of the date
- Irregular information can be derived. **Further encoding** and **preprocessing** can make it quite useful.



day

- Information of the date
- Irregular information can be derived. **Not significant** in the models

Preprocessing

Date and timestamp

- Origin dataset only contains **day and month**
- Data are **chronologically listed**
- Add feature **‘year’**

```
year_counter = 2008
ii = 0
while ii <= data.shape[0] - 1:
    if data['month'].iloc[ii] != 'jan':
        data['year'].iloc[ii] = year_counter
        ii = ii + 1
    else:
        year_counter = year_counter + 1
        for jj in range(ii, data.shape[0]):
            if data['month'].iloc[jj] == 'jan':
                data['year'].iloc[jj] = year_counter
            else:
                ii = jj
                break
```

Preprocessing

Encode categorical features: 3 examples

- **One-hot** encoder

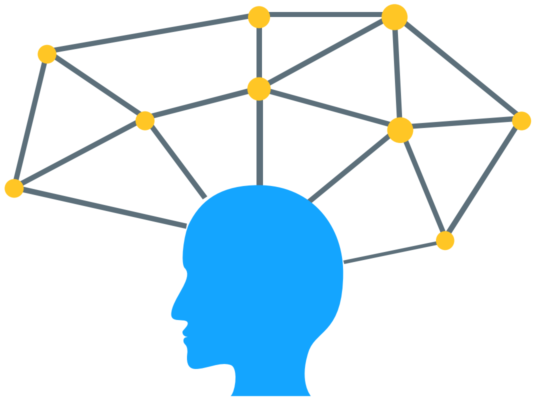
```
data = pd.get_dummies(data, columns = ['job'])
```

- **Discretization of** numerical features

```
X['pdays_0-143'] = data['pdays'].apply(lambda x: 1 if x >= 0 and x < 143 else 0)
X['pdays_143-282'] = data['pdays'].apply(lambda x: 1 if x >= 143 and x < 282 else 0)
X['pdays_282-427'] = data['pdays'].apply(lambda x: 1 if x >= 282 and x < 427 else 0)
```

- **customized** encoder

```
X['poutcome_success'] = data['poutcome'].apply(lambda x: 1 if x == 'success' else 0)
```



Model Selection
Feature engineering

Li Zitian: Random Forest vs Decision Tree

LI: Random Forest vs Decision Tree

- In terms of **Score**: Random Forest is always better *

One-hot	"year"	Smote	Generate Cross Term	RF	DT
√				**0.5588**	0.5098
√	√			**0.5897**	0.5624
√	√	√		**0.5348**	0.5100
			√ **	**0.5880**	0.5507

* Notebook is located at .\LI ZITIAN-Random Forest-Decision Tree\Model_Selcetion_Random-Forest_vs_Decision-Tree.ipynb

** We generate **300+ new features** and then use backward selcetion to drop them, and we use the best score in this process

LI: Random Forest vs Decision Tree

- In terms of **Robustness**: Random Forest depends on more features

Random Forest-100 Features

feature name	feature_importance
duration	0.5863
age	0.1124
balance	0.0872
campaign	0.0788
pdays	0.0562

Decision Tree-100 Features

feature name	feature_importance
duration	0.5758
year	**0.2610**
age	0.0956
balance	0.0430
pdays	0.0126

LI: Random Forest vs Decision Tree

- In terms of **Robustness**: Random Forest depends on more features

Random Forest-50 Features

feature name	feature_importance
duration	0.6575
age	0.1179
campaign	0.0739
balance	0.0569
year	**0.0416**

Decision Tree-50 Features

feature name	feature_importance
duration	0.5852
year	**0.2618**
age	0.0915
balance	0.0504
pdays	0.0083

LI: Random Forest vs Decision Tree

- In terms of **Robustness**: Random Forest depends on more features

Random Forest-20 Features

feature name	feature_importance
duration	**0.7352**
age	0.0880
campaign	0.0739
year	**0.0514**
balance	0.0416

Decision Tree-20 Features

feature name	feature_importance
duration	0.5922
year	**0.2693**
age	0.0903
balance	0.0328
pdays	0.0154

Feature engineering and Parameter tuning: Random Forest

Top 8 most important features

feature name	feature_importance
duration	0.6575
age	0.1179
campaign	0.0739
balance	0.0569
year	0.0416
pdays	0.0293
poutcome	0.0010
previous	0.0078

- This list consists of lots of **cross term,** which may be **hard** to explain
- To get mcc 0.58, **more than 100** features involved in the model, training time is too **long**
- **Forward selection** to deduct data
- The notebook locates at .\LI ZITIAN-Random Forest-Decision Tree\Random Forest Forward Selection.ipynb

Forward Selection: base line

- Feature List:
  [**duration**, **age**]
- hyperparmeter:
  'class_weight': {0: 1, 1: 3},
  'criterion': 'gini',
  'max_depth': 15,
  'min_samples_leaf': 10,
  'n_estimators': 100,
  'random_state':888
- Base line score: **0.38**

```
data = pd.read_csv("..\\bank-full-add_timestamp.csv",
                   sep=',',
                   engine="python")

y = data['y'].replace({"yes":1,"no":0})
data['y'].replace({"yes":1,"no":0}, inplace = True)
X = deepcopy(data[['duration', 'age']])

para = {'class_weight': {0: 1, 1: 3},
        'criterion': 'gini',
        'max_depth': 15,
        'min_samples_leaf': 10,
        'n_estimators': 100,
        'random_state':888}
rftree = RandomForestClassifier(**para)
kf = KFold(n_splits=5, shuffle = True, random_state = 888)

score_array = np.zeros(5)
for (ii, (train_index, test_index)) in enumerate(kf.split(X, y)):
#     print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X.loc[train_index], X.loc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    rftree.fit(X_train, y_train)
    y_predict = rftree.predict(X_test)
    myscore = matthews_corrcoef(y_test, y_predict)
    score_array[ii] = myscore
```

Forward Selection: day, Month, Year

	2008	2009	2010		2008	2009	2010
Jan	\	0.03	**0.46**	Jul	0.06	0.37	**0.54**
Feb	\	0.11	**0.53**	Aug	0.06	0.34	**0.52**
Mar	\	**0.48**	**0.56**	Sep	\	**0.44**	**0.49**
Apr	\	0.17	**0.59**	Oct	0.61	**0.42**	**0.41**
May	0.03	0.10	**0.57**	Nov	0.06	**0.49**	**0.48**
Jun	0.04	0.38	**0.49**	Dec	0.08	**0.49**	**0.49**

Ratio of positive label

- No information from feature 'day'
- month and day **are strongly related** to y

Forward Selection: day, Month, Year

- Encode each year and **oct in 2018**, **jan feb apr may jun jul aug in 2019**,

```
X['year_2008'] = data.apply(lambda x: 1 if x['year'] == 2008 else 0, axis = 1)
X['year_2009'] = data.apply(lambda x: 1 if x['year'] == 2009 else 0, axis = 1)
X['year_2010'] = data.apply(lambda x: 1 if x['year'] == 2010 else 0, axis = 1)
X['Encode-date_2008_10'] = data.apply(lambda x: 1 if x['year'] == 2008 and x['
X['Encode-date_2009_1_2_4_5'] = data.apply(lambda x: 1 if x['year'] == 2009 an
X['Encode-date_2009_6_7_8'] = data.apply(lambda x: 1 if x['year'] == 2008 and
X['Encode-date_2009_6_7_8'] = data.apply(lambda x: 1 if x['year'] == 2008 and
```

- mcc score jump to **0.5631**, a huge improvement

- Sep, 2008: Financial crisis in United States

  End of 2009: Debt crisis in Greek

  Exogenous factor matters!!!

Forward Selection: balance

	Treat it as numeric	Discretalize it by quantile	Multiply by duration
MCC score	0.5619	0.5569	**0.5632**

- Standard scaler is meaning less in random forest
- **Multiply by duration** is the best, but **not significant difference**
- After add it to X, our feature list:
  duration', 'age', 'Encode-date_2008_10', 'Encode-date_2009_1_2_4_5', 'Encode-date_2009_6_7_8', 'year_2008', 'year_2009', 'year_2010', 'balance'

Forward Selection: poutcome

	Success = 1, others =0	One-hot encoding
MCC score	**0.5689**	0.5681

- Four potential value of this feature: unknown, other, failure, success
- **Just encode success** is the best, have **small improvement** compared to last round
- To deduct dataset, we prefer to use more single encoding method, we add “poutcome_success” to our X

Feature engineering and Parameter tuning: Random Forest

Forward Selection: campaign

	Treat it as numeric	Discretalize it by quantile	Multiply by duration
MCC score	**0.5748**	0.5742	0.5722

- Just treat it as **numeric variable** is the best way

Forward Selection: pdays

	Treat it as numeric	encode -1	Discretalization	Multiply others
MCC score	0.5779	0.5756	**0.5799**	0.5719

- **Discretalize it by quantile** is the best way

Forward Selection: job

	Encode it by correlation of y	Multiply it by duation
MCC score	0.5797	0.5753

- This feature cannot help to imporve the score, we **drop it**

Forward Selection: housing

	One-hot coding	One-hot + Multiply it by duation
MCC score	0.5758	0.5756

- This feature cannot help to imporve the score, we **drop it**

Forward Selection: loan

	One-hot coding	One-hot + Multiply it by duation
MCC score	0.5760	0.5773

- This feature cannot help to imporve the score, we **drop it**

Forward Selection: contact

	One-hot coding	Just Encode “cellular”
MCC score	0.5810	**0.5824**

- We found that just **encode cellular is 1, while others is 0**, is the best

Forward Selection: marital

	One-hot coding	One-hot + Multiply it by duation
MCC score	**0.5866**	0.5824

- **One-hot encoding** is the best

Forward Selection: education

	One-hot coding	One-hot + Multiply it by duation
MCC score	0.5847	0.5793

- This feature cannot help to imporve the score, we **drop it**

Forward Selection: default

	One-hot coding	One-hot + Multiply it by duation
MCC score	0.5854	0.5855

- This feature cannot help to imporve the score, we **drop it**

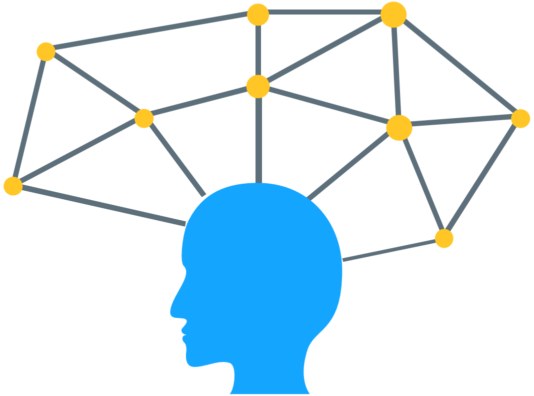
Forward Selection: previous

	Treat it as numeric	discretalize it with 0 or non-zero
MCC score	0.5855	0.5837

- This feature cannot help to imporve the score, we **drop it**

Conclusion after Forward Selection

- Best score of Random Forest: **0.5868**
  - Final feature list, remain **17** features:
    'duration', 'age', 'Encode-date_2008_10', 'Encode-date_2009_1_2_4_5', 'Encode-date_2009_6_7_8', 'year_2008', 'year_2009', 'year_2010', 'balance', 'poutcome_success', 'campaign', 'pdays_0-143', 'pdays_-1', 'pdays_143-282', 'pdays_282-427', 'contact_cellular', 'marital_single', 'marital_married', 'marital_divorced'
  - **hyper parameter** of Random Forest:
    'class_weight': {0: 1, 1: 3}, 'criterion': 'gini', 'max_depth': 15, 'min_samples_leaf': 10, 'n_estimators': 100, 'random_state': 888
- Techniques in Forward Selection
  - **Exogenuous factor** has huge impact
  - One-hot encoding may **not** be the **best encoding method** for categorical featrue
  - Discretalization of numeric feature may help to improve **robustness and higher score**



Model Selection
Feature engineering

Huang Xijie: XGBoost vs LightGBM

Data Preprocessing and Visulization for XGBoost and Light_GBM

- Visulization

- Feature Engineering

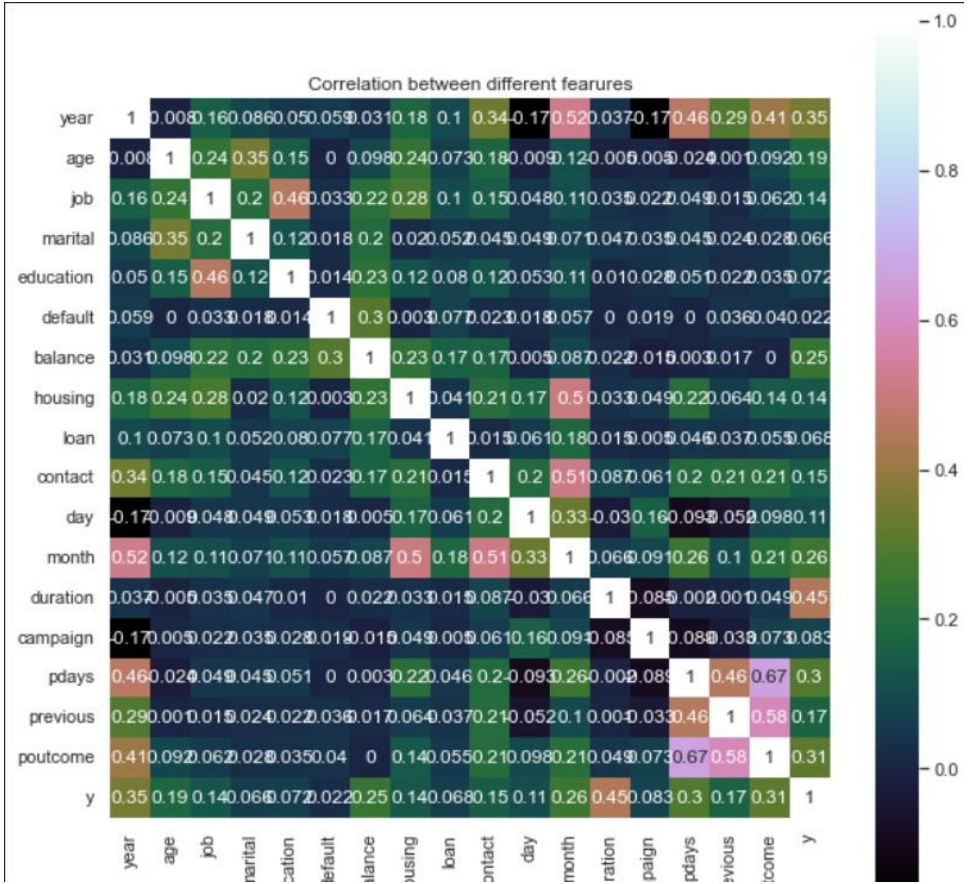
1. Data Preprocessing and Visulization

1.1 Statistics for the data

- Analysis for datatype
- Check if there are missing data
- Note that from the description in bank_name .txt, we found out that all data are ordered by dates from May 2008 to November 2010. Hence, we created a new column “Year” to help us predic the target data.

	data_type	Missing data in %	Unique	First Value	Second Value	Third Value
year	int64	0.0	3	2008	2008	2008
age	object	0.0	77	58	44	33
job	object	0.0	12	management	technician	entrepreneur
marital	object	0.0	3	married	single	married
education	object	0.0	4	tertiary	secondary	secondary
default	object	0.0	2	no	no	no
balance	object	0.0	7168	2143	29	2
housing	object	0.0	2	yes	yes	yes
loan	object	0.0	2	no	no	yes
contact	object	0.0	3	unknown	unknown	unknown
day	object	0.0	31	5	5	5
month	object	0.0	12	may	may	may
duration	object	0.0	1573	261	151	76
campaign	object	0.0	48	1	1	1
pdays	object	0.0	559	-1	-1	-1
previous	object	0.0	41	0	0	0
poutcome	object	0.0	4	unknown	unknown	unknown
y	object	0.0	2	no	no	no

Huang: XGboost vs LiaghtGBM



Correlation Matrix

- **Numerical vs Numerical**: correlaton coefficients
- **Categorical vs (Categorical & Numerical)**: **Cramer's V** method
- **Year, duration, pdays, poutcome** may have strong relationship with the outcome

Please see detail at "Data_Exploration_Visualization_Xijie_Huang" for correlation of mixed type features

1. **Data Preprocessing and Visulization**
   1.2 Feature engineering

- Do one-hot-encode for categorical data.
- **For binary entries, change “Yes” to 1 and “No” to 0, instead of doing one-hot-encode.**
- **Adding year and datetime to features**
- **Normalize the numerical data (Scaling)**

Note that the last three significantly improve the mcc score for xgboost, lightGBM and SVM respectively.

The best initial value of MCC for XGBoost and SVM are about 33% and 8%

Modelling

- XGBoost
- LightGBM
- (SVM)

2. XGBoost

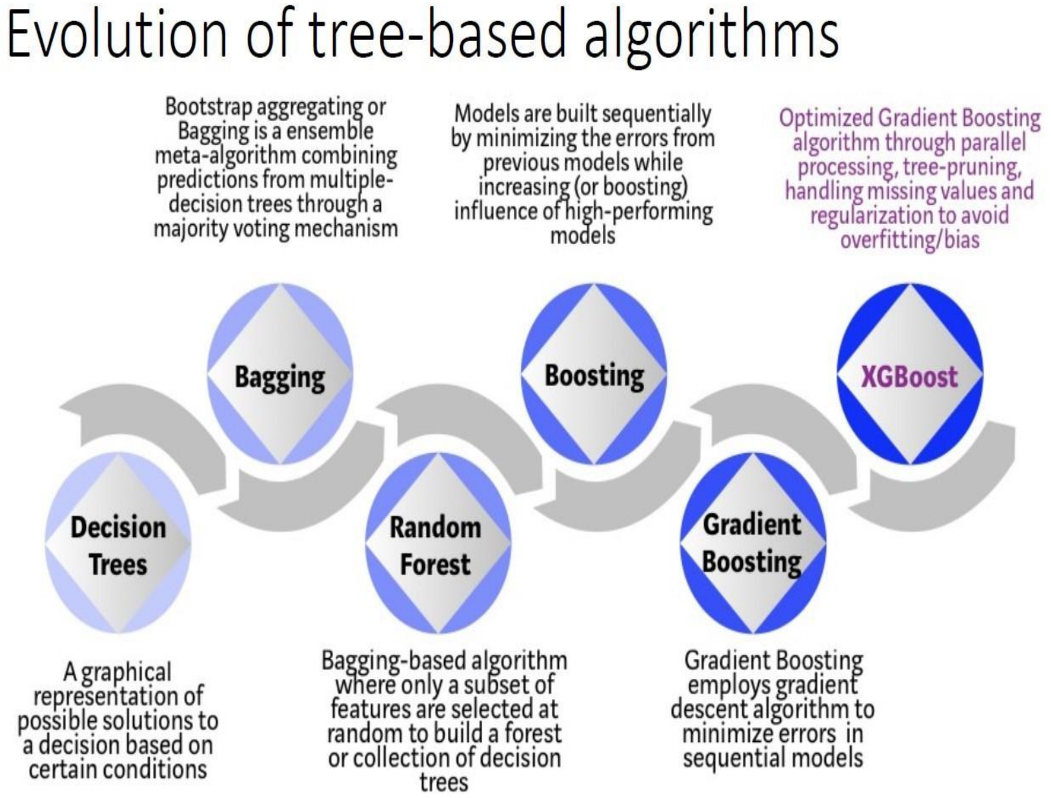
2.1 What is XGBoost

- What is XGBoost
- **XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. It is an algorithm that was developed as a research project at the University of Washington by Tianqi Chen and Carlos Guestrin in 2016**

- Why use XGBoost
- **In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms may perform better.**

2. XGBoost

2.2 Evolution of tree-based algorithms



Algorithm detail with reference: https://xgboost.readthedocs.io/en/latest/tutorials/index.html, and the original paper.

2. XGBoost

2.3 Weighted or Resample

```
model_xgb = XGBClassifier(eval_metric='mlogloss',
                          scale_pos_weight = my_scale_pos_weight,
                          use_label_encoder = False)

model_svm_ovr = svm.SVC(decision_function_shape='ovr',
                        kernel='rbf',
                        gamma='auto',
                        class_weight='balanced')

model_lightgbm = lgb.LGBMClassifier(is_unbalance = True,
                                    max_depth = 10)
```

```
mcc for XGB_classifier  0.6113463913838503
mcc for light_GBM_classifier  0.5983678791327349
mcc for SVM_classifier  0.575273057399935
Avg accuracy: 0.8865762177380392
****************************************************************
```

```
model_xgb_resampled = XGBClassifier(use_label_encoder = False,
                                    eval_metric='mlogloss',
                                    tree_method='gpu_hist')

model_svm_ovr_resampled = svm.SVC(decision_function_shape='ovr',
                                  kernel='rbf',
                                  gamma='auto')

model_lightgbm_resampled = lgb.LGBMClassifier(max_depth = 10)
```

```
(36168, 49)
mcc for XGB_classifier_resampled  0.5971537055625841
mcc for light_GBM_classifier_resampled  0.59748921061954
mcc for SVM_classifier_resampled  0.3994338283929481
****************************************************************
```

Huang: XGboost vs LightGBM

3. Light_GBM
3.1 XGBoost VS Light_GBM

- **LightGBM VS XGBoost**

To reduce the implementation time, a team from Microsoft developed the light gradient boosting machine (LightGBM) in April 2017 [8]. The main difference is that the decision trees in LightGBM are grown leaf-wise, instead of checking all of the previous leaves for each new leaf, as shown in Figs. 1 and 2. All the attributes are sorted and grouped as bins. This implementation is called histogram implementation. LightGBM has several advantages such as better accuracy, faster training speed, and is capable of large-scale handling data and is GPU learning supported.

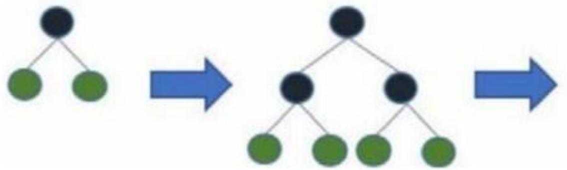


Fig. 1 XGBoost Level-wise tree growth

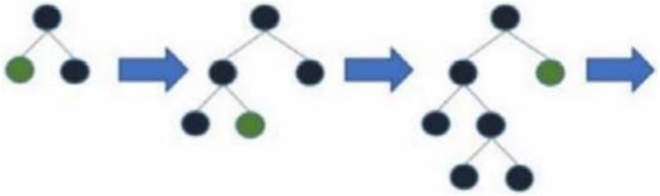


Fig. 2 LightGBM Leaf-wise tree growth

Reference: https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html#for-better-accuracy

3. Light_GBM

3.2 Efficiency or Accuracy

```
model_xgb = XGBClassifier(eval_metric='mlogloss',
                          scale_pos_weight = my_scale_pos_weight,
                          use_label_encoder = False)

model_svm_ovr = svm.SVC(decision_function_shape='ovr',
                        kernel='rbf',
                        gamma='auto',
                        class_weight='balanced')

model_lightgbm = lgb.LGBMClassifier(is_unbalance = True,
                                    max_depth = 10)
```

```
Training time for XGBoost:2.0365662574768066s
Training time for LightGBM:0.18746638298034668s
Training time for SVM:41.976288080215454s
```

```
______________________________________________________________
mcc for XGB_classifier  0.6052487615901924
mcc for light_GBM_classifier  0.6112553230801491
mcc for SVM_classifier  0.5761930063672386
Avg accuracy: 0.8865542039149725
*****************************************************************
*****************************************************************
*****************************************************************
______________________________________________________________
```

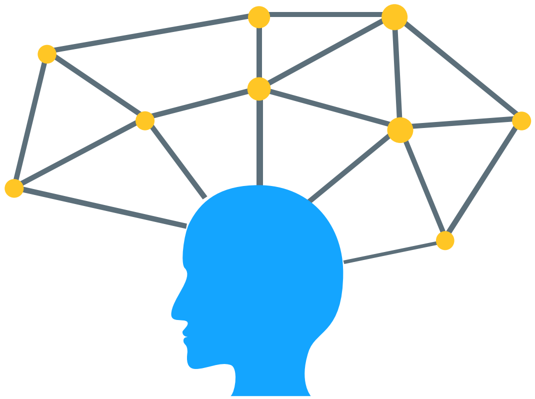
Huang: XGboost vs LightGBM

- **Feature Importance summerization**

```
Feature for XGBoost: year
Feature for XGBoost: duration
Feature for XGBoost: housing
Feature for XGBoost: month_may
Feature for XGBoost: poutcome_success
Feature for XGBoost: month_oct
Feature for XGBoost: month_jan
Feature for XGBoost: month_feb
Feature for XGBoost: month_apr
Feature for XGBoost: month_mar
```

```
Feature for Light_GBM: duration
Feature for Light_GBM: day
Feature for Light_GBM: balance
Feature for Light_GBM: age
Feature for Light_GBM: pdays
Feature for Light_GBM: year
Feature for Light_GBM: campaign
Feature for Light_GBM: month_may
Feature for Light_GBM: month_oct
Feature for Light_GBM: month_feb
```

- **Feature importance for XGBoost**
- **Feature importance for LightGBM**



Model Selection
Feature engineering

Wang Shuhui: Logistic Regression vs KNN

Wang: Logistic Regression vs KNN

- In terms of **Score**: Logistic Regression is always better

One-hot	"year"	Ordinal	Discretization	Resampling	Logistic Regression	KNN
√					**0.5097**	0.4149
		√			**0.4432**	0.3986
√	√		√		**0.5477**	0.3986
	√	√	√		**0.5448**	-
	√	√	√	√	**0.5448**	-
√	√		√	√	**-**	**0.5448**

Wang: Logistic Regression vs KNN

- In terms of **Efficiency** and **Robustness** : Logistic Regression is better

	Total number of feature used	Feature name	Feature coefficient	Resampling
Logistic Regression	**22**	**month_jan**	**-2.2789**	**No**
		campaign	-2.0008	
		year	**1.9979**	
		poutcome_success	1.8130	
		duration	1.6181	
KNN	**50 * 45 = 2250**			Yes

Base Model

- Hyperparmeters:

  ‘solver’ :'newton-cg',

  class_weight: ‘balanced’,

  ‘C': 10

- Base line score: **0.5097**

```
for c in C:
    log_clf = LogisticRegression(solver='newton-cg', class_weight='balanced', C = c)
    log_scores = []
    for train_index , test_index in kf.split(x_oh, Y):
        X_train, y_train = x_oh.iloc[train_index], Y.iloc[train_index]
        X_test, y_test = x_oh.iloc[test_index], Y.iloc[test_index]
        log_clf.fit(X_train,y_train)
        pred_values_log = log_clf.predict(X_test)
        log_score = matthews_corrcoef(pred_values_log , y_test)
        log_scores.append(log_score)
    avg_mcc_score_log = sum(log_scores)/5

    print('---LogisticRegression---')
    print('Inverse regularization strength - {}'.format(c))
    print('accuracy of each fold - {}'.format(log_scores))
    print('Avg accuracy : {}'.format(avg_mcc_score_log))
```

Feature engineering and Parameter tuning: Logistic Regression

Top 6 Important Features

Feature name	Feature coefficient
age	0.1000
marital	0.2528
month	0.0313
duration	1.3999
campaign	-0.4268
previous	0.3350
poutcome	0.1975

- In base model, 48 features were involved. Adapting **ordinal encoded inputs** will reduce computational cost significantly.
- This list consists of coefficients of inputs that show preference towards one of the output classes in base model fitted with ordinal encoded data.
- To get higher MCC score, **preprocessing** on these features is necessary.

Adding New Feature: Year

- Instead of ‘year’ = ‘2008’, ‘2009’ or ‘2010’, we apply ordinal encoding to attribute ‘year’.
- MCC score increases from **0.4432** to **0.5056**

```
# Adding new feature
row_sum = X.shape[0]
X_year = np.zeros(row_sum, dtype=int)
year_counter = 0
new_year = False
for i in range(row_sum):
    if X.loc[i,'month'] == 'jan':
        if not new_year:
            year_counter = year_counter + 1
            new_year = True
    else:
        new_year = False
    X_year[i] = year_counter
```

Discretization: poutcome

	Success = 1, others =0	Ordinal encoding
MCC score	**0.5190**	0.5056

- Four potential value of this feature: unknown, other, failure, success
- **Just encode success** has best score **improvement** which also matches the histogram plot of ‘poutcome’

Discretization: Age

	Treat it as numeric	Discretalize into 3 classes
MCC score	0.5190	**0.5209**

Discretization: previous

	Treat it as numeric	Discretalize it into 2 bins
MCC score	0.5209	**0.5214**

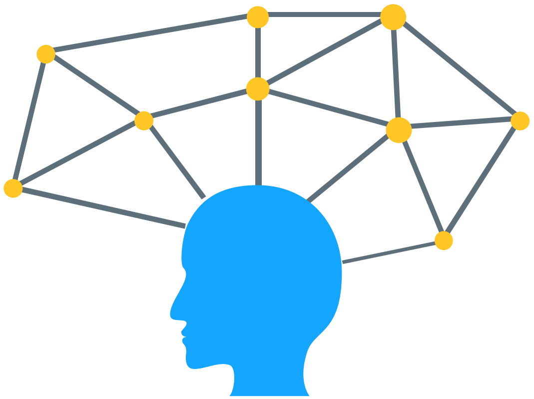
Feature Selection: Month

	Ordinal encoding	Keep only 6 months	One-hot encoding
MCC score	0.5214	**0.5448**	0.5478

- Using one-hot encoded month features produces the highest prediction accuracy.
- This will increase input feature dimension to about twice the original size.
- Dropping less significant feature columns (['apr','aug','feb','jul','nov','sep']), will **maintain the MCC score** with just **6 additional input columns**.

Conclusion after Feature Selection

- Final score of Logistic Regression: **0.5448**
  - Final feature list, remain **22** features:
    'age', 'job', 'marital_married', 'education', 'default', 'balance', 'housing', 'loan', 'contact', 'day', 'duration', 'campaign', 'pdays', 'previous', 'poutcome_success', 'year', 'dec', 'jan', 'jun', 'mar', 'may', 'oct'
  - **hyper parameter** of Logistic Regression:
    'class_weight': 'balanced', 'solver': newton-cg, 'C': 10, max_iter: 1000, 'fit_intercept': True
  - **New feature 'year'** has huge effect in score improvement
  - Discretalization of numeric feature and customised encoding help to enhance **model performance**



Summary & Insights

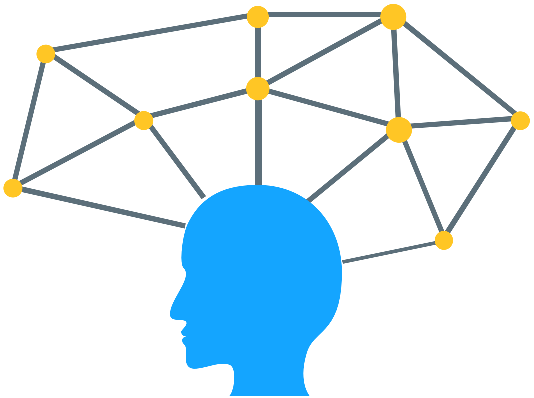
Model Achievements

Model	Number of features invovled	Model Training Time	MCC score
XGBoost	16	Short	0.60
Random Forest	17	Long	0.59
Logistic Regression	22	Shortest	0.54

Insights

- Techniques in **model selection**
  - Some algorithms **may always perform better** than other one, at the cost of longer training time
  - Hyper parameter have **large impact** on the performance of algorithms. Finding the best hyperparameter may **take more time** than feature engineering.

- Techniques in feature engineering
  - One-hot encoding is **not the best encoding scheme** for all categorical featrues
  - Discretalization of numeric features help to improve model **robustness** and have **higher mcc score**



Future Direction

Future Direction

- Additional Features
  - We noticed that feature “year” is quite important in our study. This feature may contains **key information of economical environment**
  - If we want to identify clients who are more willing to make term deposits, we need to **fix economical factor.** Or our analysis may not be useful
  - Note that the feature “**duration**” is also important. It is natural that those who are **willing to invest** are more likely to **spend more time** to discuss about the detail of **investment during the last contact.** So more features related to duration may be useful.

Future Direction

- Parameter Tuning
  - Due to the limitation of time and computing resource, **more powerful computing resources is required** to carry out grid search
  - Parameter tuning and feature selection should be **repeatedly tested together**. More rounds of feature engineering may help to get better score.
- Feature Engineering
  - There are different methods to encode features. Sometimes **customized encoding rule** may have **better** performance than one-hot encoding
  - Discretalizing numeric features **by quantile** may be a good choice, but it is not applicable if we just want to have two bins.

Thank you!